NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

QUANTUM COMPUTERS
AND
THEIR IMPACT ON DoD IN THE 21st CENTURY

by

John E. Mades

September 1999

Thesis Advisor: Neil Rowe
Thesis Co-Advisor: Wolfgang Baer

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1999	Master's Thesis

4. TITLE AND SUBTITLE
QUANTUM COMPUTERS AND THEIR IMPACT ON DoD IN THE 21ST CENTURY

5. FUNDING NUMBERS

6. AUTHOR(S)
Mades, John E.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Computer processor speeds double every eighteen months according to Moore's law. This growth will reach a limit by the year 2020. Quantum computation is one proposed alternative to bypass this limitation. This thesis explores the topic of quantum computation. Specifically, we address what is a quantum computer, its various proposed implementations, its technological feasibility, and its military applications. Recent experiments have provided a proof of concept for quantum computation and some researchers believe that a working model could be developed within a reasonable time period. This success has caused a marked increase in the interest in quantum computers and their proposed potential. We attempt to separate fact from fiction to see what possible benefits the Department of Defense could obtain from it.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Quantum Computers, Quantum Parallelism, Teleportation, Quantum Communication, Quantum Cryptography	83
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI- CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited

QUANTUM COMPUTERS AND THEIR IMPACT ON DoD IN THE 21ST CENTRUY

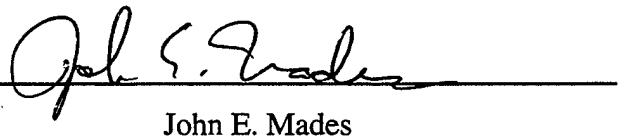
John E. Mades
Captain, United States Marine Corps
B.S.,University of Houston, 1990

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

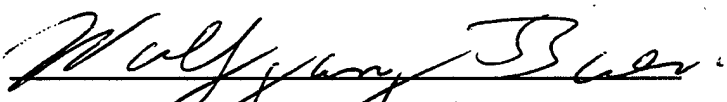
NAVAL POSTGRADUATE SCHOOL
September 1999

Author: 

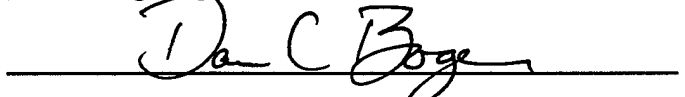
John E. Mades

Approved by: 

Neil C. Rowe, Thesis Advisor



Wolfgang Baer, Co-Advisor



Dan Boger, Chairman
Department of Computer Science

ABSTRACT

Computer processor speeds double every eighteen months according to Moore's law. This growth will reach a limit by the year 2020. Quantum computation is one proposed alternative to bypass this limitation. This thesis explores the topic of quantum computation. Specifically, we address what is a quantum computer, its various proposed implementations, its technological feasibility, and its military applications. Recent experiments have provided a proof of concept for quantum computation and some researchers believe that a working model could be developed within a reasonable time period. This success has caused a marked increase in the interest in quantum computers and their proposed potential. We attempt to separate fact from fiction to see what possible benefits the Department of Defense could obtain from it.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Moore's law provides a rough estimation for the extrapolation of computer processor speeds over time. Moore's law states that "Computer processor speeds double every eighteen months". But there is a limit: Reference 1 asserts that with the current technology the physical limit for processor speeds will be reached in the year 2020. The current technology relies on placing more transistors in a smaller area on the microprocessor surface in order to increase computing speed (clock speed). Concomitantly there is a heat dissipation problem due to the increased number of transistors per unit area. Therefore by extrapolating the energy dissipation trend the current technology should reach the thermal noise limit for the atomic level around the year 2020 [Ref. 1]. At this point we will have entered the quantum realm. Additionally, there is an economic concern in that the cost of constructing "clean" rooms to produce semiconductors doubles every three years [Ref. 1, p. 11]. At this rate, the cost of producing a new semiconductor plant in 2020 will reach the trillion-dollar level. Clearly this would be prohibitive to business. What do we do then? Quantum computation is one of the areas being explored to answer this question.

How does this pertain to us in the Department of Defense? The military mirrors society in many aspects, one of which is the ubiquitous use of computers. We in the military currently use computers for everything from administration to target guidance. The Joint Vision 2010 [Ref. 21] points out, "This era will be one of accelerating technological change. Critical advances will have enormous impact on all military forces. Successful adaptation of new and improved technologies may provide great increases in specific capabilities. Conversely, failure to understand and adapt could lead

today's militaries into premature obsolescence and greatly increase the risks that such forces will be incapable of effective operations against forces with high technology." Therefore we must examine new areas of technology that could represent a significant benefit to our warfighting capability. Quantum information systems is an area of new technology that could provide that benefit and the potential payoff is enormous. The arena of information systems is one of the pillars of our warfighting ability and one of our greatest vulnerabilities. We depend upon the unfettered access to the electromagnetic spectrum, whether through electromagnetic waves or wires. However, DoD computer systems endure hundreds of "cyber" attacks every day. To continue our quest for the "high ground" we need to explore new technologies as they are being discovered.

Applications for DoD cryptographers could include quantum cryptography, new ways to break and make codes, which with the explosion of the Internet could help better secure the safety of DoD computer systems. Additionally, quantum computers can offer a solution to these needs. Through "quantum parallelism" the Schor algorithm for quantum computation may be able to assist in breaking some of today's unbreakable codes or developing stronger ones. "Quantum parallelism" may allow us to build neural nets that can learn more quickly and may allow us to solve some of the more interesting computation problems known as NP-hard/complete.

Imagine a radio intercept system that collects over the entire electromagnetic spectrum, simultaneously decrypting and providing real-time scanning for key words or phrases could be possible. Imagine a computer system that allows artificially intelligent systems to process exponentially faster because it can reason millions of ways in parallel. Cryptographic key exchanges could be 100% secure with the ability to detect

eavesdroppers. The possibility of these conjectures was relegated to science fiction until recently due to limitations in computer speed and system "knowledge".

Isaac Chuang's recent experiment using Nuclear Magnetic Resonance (NMR) techniques to apply Grover's Algorithm has generated increased interest in the realization of quantum computers. In the fall of 1997 when our group began preliminary research into quantum computers, a search of the Internet found maybe 20 sites pertaining to quantum computation. One year later there are over 1,800 sites. This shows the remarkable interest in the potential for quantum computing. Additionally, as noted in Ref. 2, quantum mechanics provides a better explanation of physical reality than classical mechanics and can therefore provide more information about the "complexity" or "knowledge" of the system [Ref. 8, p. 97].

However, this interest should be tempered with a sense of caution. The results of quantum mechanics are well documented experimentally, but academic understanding will not automatically yield an engineering success. There are several impediments to the construction of a quantum computer: decoherence, size, economic factors etc. Therefore we must proceed toward the goal of constructing a quantum computer with those impediments in mind.

This paper will explore quantum computing by explaining what a quantum computer is, why we should choose to build one, how we could implement one, and its limitations and advantages for military needs.

II. BACKGROUND

In quantum mechanics, a system can be interpreted as existing in a superposition of states simultaneously, such as a bit of information being both a 1 and a 0 at the same time. This superposition of states is the key to the possible exponential increase offered by computing at the quantum level. Each of these states can separately perform calculations for the solution of a problem. However a theoretical impediment to quantum computing is the lack of a machine model. The Turing model of computation, developed by Alan Turing, appears inadequate to describe the computational processes occurring. In the Turing model, questions of how to implement the reading of the data tape and writing to the data tape are ignored, essential questions that cannot be ignored under quantum computing. Therefore, a new model must be developed to describe and formalize the process. This new model should be based on applied mathematics or physics in order to more accurately give insight into the quantum computational process. No longer can we think of data as just 1's and 0's, but as the probabilities of a 1 or 0, representing superposition of states. Additionally, the quantum computer can no longer be thought of as a tape head and an infinitely long tape. In accordance with quantum mechanics we must think of the tape as a system described by the state function $\Psi(t)$ that evolves over time to perform the calculation.

Classical computer systems perform computation by sending an electric signal through a circuit in conjunction with a timing signal. This signal is independent in that it does not require interaction with any other signal in order to perform its calculation. Calculations performed in this manner take place through space (the microprocessor) and time (the length of time required to traverse the circuit). A quantum computer system

performs its calculations by controlling the memory evolution over time. Quantum computations must be prepared in an initial state, which would correspond to "input". This input is then transitioned to other quantum states by one of several methods (see table 2 in section III for a list of methods). Input to a general quantum computation is now more than just the concatenation of its bits, because each input bit can be "entangled" with its neighbor and each bit is in a superposition of states. Entanglement is "a property of correlated pairs of quantum systems"[Ref. 1, p.180]. Two elements can be entangled if they are emitted by the same source at the same time as in the case of photons in an Einstein-Podolsky-Rosen (EPR) experiment. A good model of entanglement is a linear system of four blocks attached to each other by springs while the springs are in tension. A block in this system cannot be moved without influencing its neighbors. In this way calculations are dependent not only upon the evolution of the system in time, but the interaction of the input bits as they evolve over time.

Classical computer systems have an understood and well-defined computational model known as the Turing model. This model was developed independently by four men, Alan Turing, Alonso Church, Kurt Gödel, and Emil Post, and is a *mathematical* model for the computing process. The Turing machine consists of an infinite "tape" in two directions with a "head" that is capable of reading and writing to the "tape". The tape is prepared such that data and a "program" (set of instructions) are resident on the "tape". The "program" or set of instructions is run on the machine using a finite-state machine model that allows it to read and write to the "tape" and then halt. The tape is pulled past the tape head to allow it to read and write to the tape. This model provided the insights necessary for the construction of the modern "universal computer" like the

personal computer that sits on your desk at work, a machine just as capable of doing your word processing as it is at playing a video game.

The Turing model is discrete and works well when we are working with a classical interpretation of physical reality. However, questions regarding how and when one reads and writes to the "tape" cannot be ignored with a quantum computer since if we "look" at the tape before the computation is finished we have altered the answer, according to quantum mechanical theory (Heisenberg uncertainty principle). Therefore, when we implement a quantum computer we can try to simulate it with a classical system or choose a physical system that can change over time in order to provide us with the desired calculation. The first case is impossible and the reasons are explained in reference 2. However, with the second we must be careful in our selection of the physical system because not every physical system will provide a meaningful calculation or be "universal". An uninteresting physical system for calculation is a brick. If the brick is elevated off the ground and then dropped we could consider that a calculation of its trajectory; however, it is not scalable to our current technology standards and is not expandable to do other calculations.

We discuss now two seminal papers that form the foundation of study in this field, references 2 and 8.

A. FEYNMAN'S APPROACH

Feynman's paper [Ref. 2] is concerned about simulating physics with computers and trying to create a simulator for quantum mechanics. Therefore, he does not address the "universal quantum computer". Feynman describes the general attributes that are necessary for a computer to simulate physics correctly as "...that there is to be an exact

simulation, that the computer will do exactly the same as nature." The stipulation of an exact simulation is understood to be a fidelity measure of the system: We should expect the same results from the computer as from nature. Additionally, he points out that "The rule of simulation that I would like to have is that the number of computer elements required to simulate a large physical system is only to be proportional to the space-time volume of the physical system." He would consider simulation a failure if in order to simulate physics he would have to build a computer that would be excessively large. Also the number of elements should not increase exponentially with the given problem and should not take the lifetime of the universe to solve.

Feynman then makes the point that computers imitate time rather than simulate it, with discrete state transitions. If a computer is to simulate quantum-mechanical processes, it must simulate probability. Discrete systems can accomplish this through various means, one of which is modeling it with Markov chains. A Markov chain is a way of modeling a stochastic process by assuming that the system can be in one of a finite number of states; the probability of being in a state is dependent only on the state that immediately preceded it in time and not any of the other states that it had to pass through to get to the current state. A Markov process would not produce the kind of fidelity that Feynman is referring to because the information in it is less than the information representible in a quantum one. Feynman uses the example of a system with R particles, where probabilities must be assigned for each particle's position x such that each particle has one of a finite set of positions x_1 , x_2 , ..., x_R at time t. Then a k digit number can be assigned as the probability for the given particle at that position at that time. One problem with this model is that one must ignore the probability that some

occurrences might happen, for example: "If the probability of something happening is 10^{-700}, we say it isn't going to happen" [Ref. 2, p. 472]. The second problem is you would need a separate k-digit number for R particles in every configuration of the system and for all N points in space. The ability to store and simulate this much information is beyond the capability of classical computers currently; therefore, he concludes that a direct simulation is not feasible. However, the task may be handled by a system which imitates nature. The two choices that he presents are a probabilistic classical computer or a natural system that can simulate the computation. The answer he provides for utilizing a probabilistic classical computer is a definitive no because classical systems cannot replicate the true randomness found in nature. Therefore we should simply observe an imitative system and record the answer. Feynman does not offer a specific system or design for a universal quantum simulator; rather he states that "we should try to find out what kinds of quantum mechanical systems are mutually intersimulatable, and try to find a specific class, or a character of that class which will simulate everything". Although Feynman's tone is somewhat doubting, he later wrote a follow-up paper [Ref. 9] to revise his initial thoughts here and expound on the ideas brought out in David Deutsch's paper.

B. DEUTSCH'S CONTRIBUTION

In 1985, from a different perspective than Feynman, Deutsch approached quantum computing through the Turing or Church-Turing hypothesis [Ref. 8]: "The assumption that the intuitive notion of 'computable function' can be identified with the class of partial recursive functions..." [Ref. 10, p. 166]. Deutsch paraphrases this as: "'Every 'function which would naturally be regarded a computable' can be computed by the universal Turing machine" [Ref. 8, p. 99]. He refers to this as the weak form of the

Church-Turing hypothesis, because the language does not reflect concepts that are easily definable for physical systems. So he restates the hypothesis as the Church-Turing *principle*: "Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means" where "a computing machine M is capable of perfectly simulating a physical system ϑ, under a given labeling of their inputs and outputs, if there exists a program $\pi(\vartheta)$ for M that renders M computationally equivalent to ϑ under that labeling. In other words, $\pi(\vartheta)$ converts M into a 'black box' functionally indistinguishable from ϑ." [Ref. 8, p. 99]. Computational equivalence is important to computer science because that is where we derive the "universal computer", a machine that can compute any function that a Turing machine can compute. Deutsch says that both a Turing machine and a quantum computer operate by finite means: "(i) only a finite subsystem (though not always the same one) is in motion during any one step, and (ii) the motion depends only on the state of a finite subsystem, and (iii) the rule that specifies the motion can be given finitely in the mathematical sense (for example as an integer)." [Ref. 8, p. 100]. He shows that there is no difference between a machine that alters the input state (classical computer method) in order to calculate/compute and one that alters its constitution to become a different machine computing a different function (quantum computer method). Thus Deutsch establishes a case for computational equivalence between a Turing machine and a universal quantum computer.

Deutsch argues that it is not physical laws which inhibit us from "computing" certain functions, but the algorithms designed for the classical systems. Therefore, if we were to develop a universal quantum computer and the algorithms that utilize it then we

might expand our pool of "computable" functions. A "universal quantum computer" could simulate "ideal closed (zero temperature) systems, including ... quantum computers and quantum simulators, with arbitrarily high but not perfect accuracy." [Ref. 8, p.102]. So Deutsch claims that perfect accuracy is not required because he is investigating the quantum simulator not quantum computer. Deutsch provided the impetus to expand this area of research to its present state with this comment.

C. BELL'S THEOREM

Quantum computers are only one area of research in quantum information systems. Other areas include quantum communication and quantum cryptography, which use Bell's inequality.

When quantum theory was developed, several prominent scientists disagreed with a part of the theory known as *quantum nonlocality*. The problem is this: "Classical physics holds that any relationship existing between two or more particles must be mediated by a local force-attraction, repulsion, or at least some signal. Quantum mechanics predicts that two particles can be correlated instantaneously in the *absence* of any such force or signal..." [Ref. 11, p. 65]. Einstein together with Podolsky and Rosen (EPR) argued that a thought experiment developed by David Bohm to test the validity of quantum nonlocality proved it impossible. The thought experiment is formulated as two particles shot off from a common source toward opposite sides of a room. According to quantum mechanics the spins of the two protons, when measured, should be instantly correlated-- that is, when the spin of particle A is found to be "up", the spin of particle B should be found to be "down", at exactly the same moment. If the measurement of proton A can produce an instantaneous effect on a distant proton B, then it should be

possible to detect an intruder "wiretapping" a communication channel built from a stream of protons A. Spin is essentially a measurement of the angular momentum of a particle and is inherent in sub-atomic particles. Einstein saw this as a paradox because in order for B to "know" what A's spin was (under classical physics) a signal would have to travel between A and B instantaneously. This is not allowed under relativity (nothing can travel faster than the speed of light). Bell showed that there is no paradox because this cannot be explained by classical physics, only by the laws of quantum mechanics. Intuitively, we can say that the two spins are "entangled" to begin with and thus no information need be transmitted later between A and B.

III. WHAT IS A QUANTUM COMPUTER?

To understand what a quantum computer is, we will start with a common reference point: a personal computer. "Intuitively, a computing machine is any physical system whose dynamical evolution takes it from one set of 'input' states to one of a set of 'output' states. The states are labelled in some canonical way, the machine is prepared in a state with a given input label and then, following some motion, the output state is measured." [Ref. 8, p. 97]. We can illustrate this idea in the figure below.

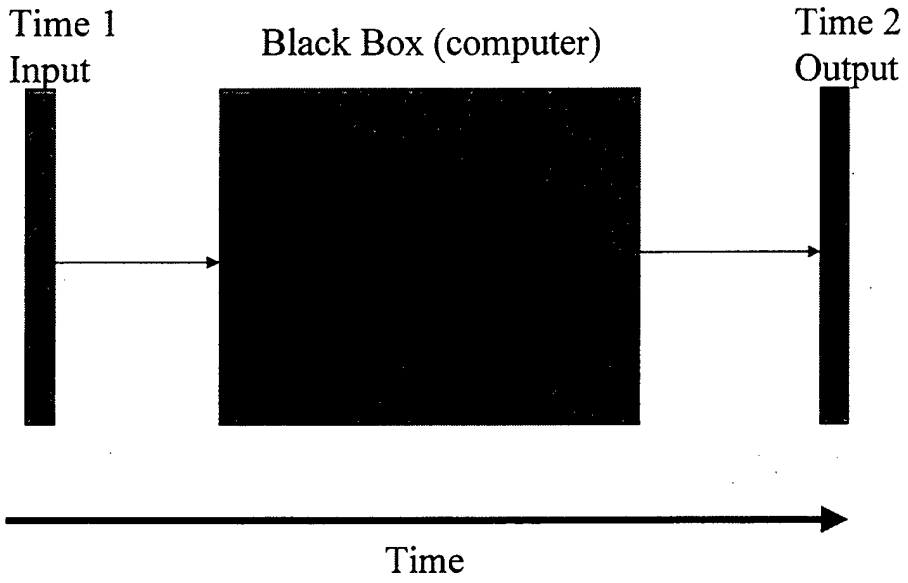


Figure 1

A. PROPERTIES OF QUANTUM COMPUTATION

There exists a computational equivalency between this model and a quantum computer. In both cases the computation is considered to be generated by the time evolution of a computer memory from an initial to a final state. In a quantum computer it

is not the state of the memory but the probability of measuring a state that is propagated in time. However, this is only the beginning. To design a quantum computer we must add mechanisms to implement superposition, entanglement, logical/physical reversibility, coherency, time-independence and output interrogation.

1. Superposition

In a classical computer one bit of information is stored as either a certain 1 or a certain 0. In quantum systems, on the other hand, one "qubit" of information will reside in a superposition of states as a 1 or 0 or both, before measurement. If we measure the system we collapse the wave function ($\Psi(t)$) and the most likely answer to that point in time will be the result of our measurement. Before measurement, a qubit register n qubits long represents 2^n states of information. This is a substantial increase over a classical system, and can be especially powerful if the qubits are correlated or "entangled". Superposition is a necessary but not sufficient condition in and of itself to derive the benefits of quantum computation. We can perform a calculation with data that has been placed in a known superposed state. However, this is performance of a classical calculation with a quantum system, because once the calculation is complete and we measure the result we only get the single value of that qubit at that time rather than the average of all the values that that qubit had been.

2. Entanglement

An additional requirement for quantum computation is correlation or "entanglement" of the qubits. "Entanglement allows one to encode data into non-trivial multi-particle superpositions of some preselected basis states" [Ref. 23, p.1]. Stated another way, entanglement provides the additional condition necessary for quantum

computation because it is the interaction of the qubits through time *and* superposition that provides the "quantum parallelism". Quantum parallelism allows us to perform calculations on the 2^n states simultaneously. When we measure a superposed and entangled quantum system we get one set of the values that the qubit(s) had been after the calculation. Obtaining more than two entangled particles is very difficult, however.

3. Logical/Physical Reversibility

The condition of logical reversibility refers to an observer's ability to infer the input from a given output. A stipulation of maintaining a quantum system is that it be physically reversible, which requires that the wave function not be collapsed due to the physical process (qubits passing through a quantum gate). Physical reversibility is required because the fundamental behavior of nature is time-reversible. Time order is impaired by measurement, which also increases entropy and looses information. If a quantum calculation is to proceed without the loss of information, the calculation must be carried out in a physically reversible system. Physical reversibility also says that when a quantum element (qubit) undergoes a physical interaction, that interaction cannot change the time-independence or coherence (a concept explained shortly) of the quantum element. The only exception to the condition of reversibility in a quantum computation is the final step (when we read out our answer). A way of ensuring this is to make quantum gates and quantum circuits logically reversible. The table below illustrates a logically reversible controlled XOR exclusive-or.

Table 1

Input #1	Input # 2	Output #1 (Answer)	Output #2 (control bit)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

As we can see from the table if the inputs are the same then the "answer" is 0, if the inputs are different then the "answer" is 1. Thus we can infer input 1 from output 1 and output 2 through another XOR operation: (output 1 XOR output 2 = input 1). Output 2 is the control bit for the operation and is the same as the input 2; therefore we have retrieved the input from the output and the operation is logically reversible. The Toffoli gate is an example of a quantum gate that performs a similar type of calculation; however it has three inputs and three outputs.

The extent to which a system must be reversible has not been fully explored. It is possible that quantum error correction can compensate for irreversibility. Classical error correction uses extra bits incorporated into a signal to determine bitwise parity deviations from the original signal. Similarly, quantum error correction could use extra qubits to determine errors in a set of qubits. Notwithstanding, it is doubtful whether quantum error correction can compensate for both losses due to coherency and losses that would naturally occur due to qubit interactions with quantum gates.

4. Coherency

Coherency losses are the major problem for a practical quantum computer. Coherency is the absence of a coupling between a quantum system and its environment. "Decoherence is more insidious. Rather than a gross bit flip, decoherence is a coupling between two initially isolated quantum systems (the qubits and its environment) that tends to randomize the relative phases of the possible states of the memory register." [Ref. 1, p.215]. Solving decoherence means to effectively isolate the quantum system from its environment. This coupling causes a quantum-mechanical effect to lose its quantum properties and become a macroscopic event. Some techniques currently employed to reduce decoherence include thermal isolation (low temperatures from cryogenic cooling or other methods), electromagnetic isolation, and radiation shielding (from cosmic rays and other interference). Perfect coherence is not an attainable goal because we can never fully isolate a given system from all effects; therefore we try to limit those effects during a given time period in a given space.

5. Time-Independence

Time independence means that the calculating elements do not change over time. In a classical system a computational element can perform different operations at different times because the calculation steps are transitions between distinct system states. However, within a quantum calculation we cannot be so definite. For example, can a quantum gate be an XOR gate at time t_1 then at time t_2 a NOT gate? We think not. The reason is that we cannot be sure at what time the qubit leaves the gate, so we cannot "know" when to change the gate to something else. We cannot be sure when the qubit "leaves" the gate because it is in a quantum-mechanical state and has a probability of

being everywhere at anytime within the system. If quantum gate arrays are required to be time-independent, we cannot use switchable circuits to perform quantum calculations.

6. Output Interrogation

Once the quantum calculation has been performed, we are left with an answer that resides in a superimposed and entangled state. We could directly measure this state. That measurement would yield one of all possible answers of the calculation and does not utilize the full potential of quantum computing. Instead, a major benefit of quantum computing is its ability to extract some overall property of the calculation (average, frequency etc.)[Ref. 24]. To extract this property we must have an output interrogator. This interrogator allows us to sample the quantum answer in a time-independent manner. This permits us to preserve the quantum nature of the answer and extract the overall property that we are interested in. We can then display the result from the output interrogator in histogram form for us to interpret.

B. A GENERAL QUANTUM COMPUTER MODEL

The following figure represents a model for a quantum computer which calculates the following function.

$$\sum_{0}^{2^m-1} \sum_{0}^{2^n-1} f(x, y, a, b, c)$$

Equation 1

where x and y are variables and a, b, and c are constants. Classically this would require $(2^m - 1)(2^n - 1)$ calculations. A quantum computer would require only one. Here is how this calculation is done.

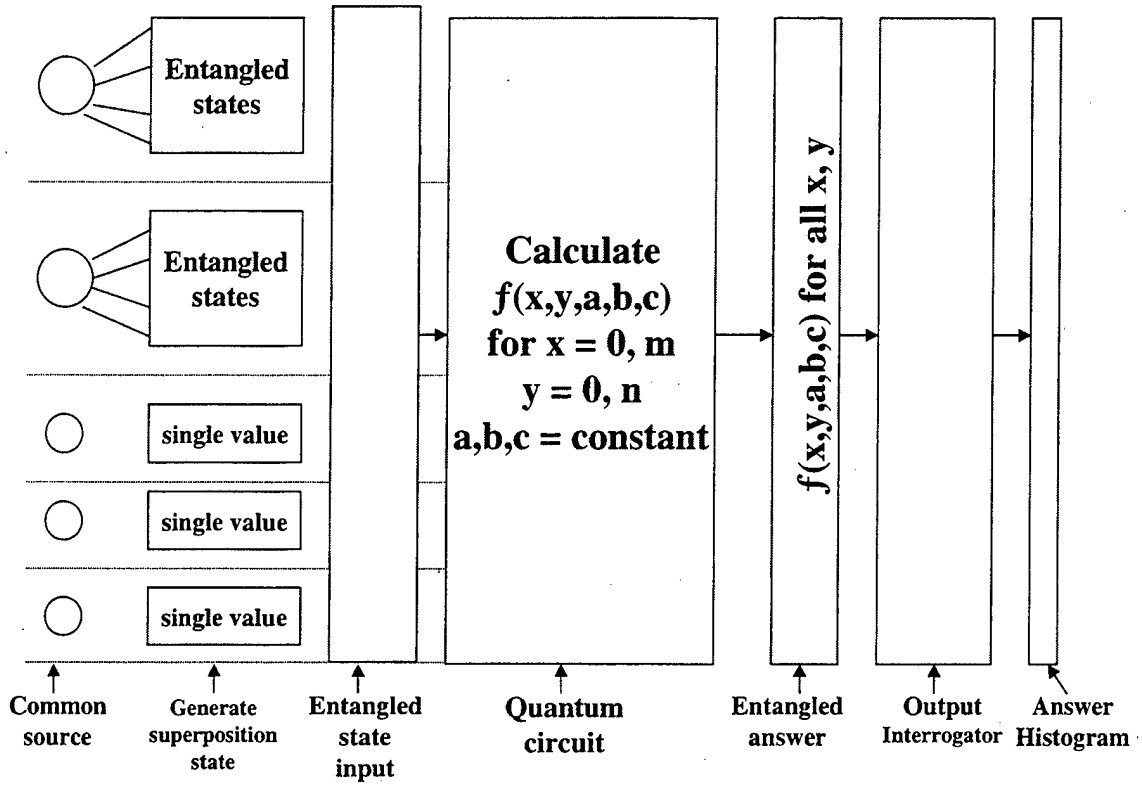


Figure 2

A common source emits entangled particles that are then put into a superposed state. At the "entangled state input" position, quantities x and y have been transformed into qubits and have $(2^m - 1)$ and $(2^n - 1)$ possible values respectively. We can perform a calculation with these qubits with some quantum circuit. Let us say that we calculate $f(x, y, a, b, c)$. Then the circuit performs the calculation on all the superposed states at the same time by transforming the wave function $\Psi(t)$ into another wave function $\Psi'(t)$ while preserving logical/physical reversibility, coherency and time independence. The completed calculation is shifted to the "entangled answer" register. We now interrogate this answer according to what overall property of the function we are looking for (average, frequency etc.). In the case above we need a summation and the output interrogator is constructed

for that purpose. The output of the interrogator is the overall property of the function we want to calculate and is displayed in histogram form. In an ideal machine only one answer will appear. In practice the histogram is generated instead of a single answer because noise and other calculation errors will broaden the single answer peak.

IV. REALIZATIONS

A. QUANTUM GATE MODEL

Digital computers process information by routing bits (signals or voltages) through switches or gates. These gates can be thought of as rudimentary logical operations. Typical gates are the AND, OR, NOT, NAND, NOR and XOR (exclusive-or).

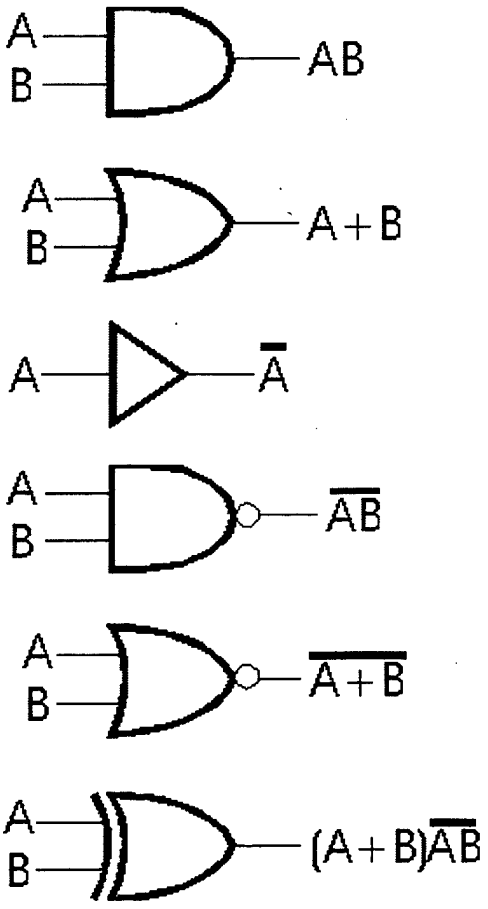


Figure 3

By interconnecting these gates, their operations can be strung together to form more complex operations such as addition and multiplication. It is the bit/signal/voltage that is routed through the gate.

Conversely, in a quantum system the gate is passed through the data, and it is the interaction between the bits (qubits) that is important. Additionally, the number of inputs

must be equal the number of outputs for a quantum gate because otherwise information would be irreversibly destroyed and the input cannot be inferred from the output. Two proposed exceptions are the FANOUT and ERASE gates pictured below.

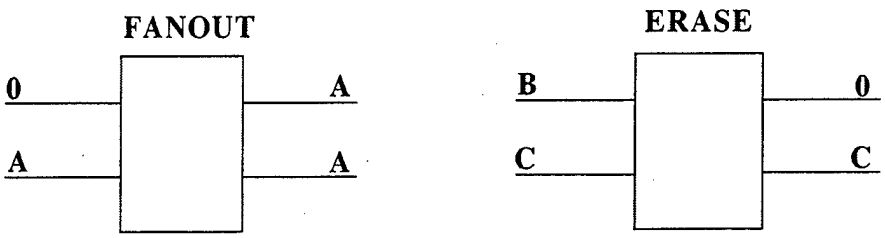


Figure 4

Here both the FANOUT and ERASE have the number of inputs equal to the number of outputs, but one input for the FANOUT is a constant and one output for the ERASE is constant. It is thought that the FANOUT gate cannot be constructed because it would require the copying of the information contained in the quantum state which is not possible. The ERASE gate on the other hand is possible because destruction of information occurs naturally as a result of decoherence and other effects.

The following figure illustrates a Toffoli quantum gate, one of the universal gates proposed for a quantum computer.

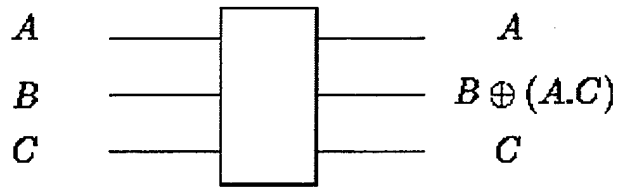


Figure 5

As the qubits pass from left to right into the gate along "quantum wires" the gate performs an elementary logical operation on the qubits. Qubits A and C pass through the gate unchanged, while B is an XOR of qubits A and C. The gate is configured with three inputs and three outputs because Toffoli originally reasoned that a workspace (scratchpad) bit was required when the gate calculation was done. Reference 22 provides a strong argument that this workspace bit is not required and the controlled-XOR discussed in chapter III is sufficient.

B. EXAMPLES OF QUANTUM GATES

With these specifications in mind, can any implementations provide us with a quantum gate? Here is an example [Ref. 13].

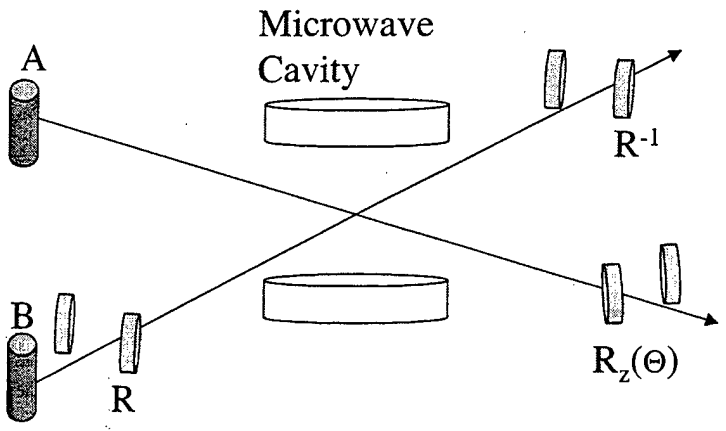


Figure 6

The figure is of a quantum electrodynamic (QED) cavity. Sources A and B supply two-state particles that are qubits. The qubits are encoded by spin, meaning that a spin "up" particle is a 1 and a spin "down" is a 0. A particle is passed from A through the cavity while it is in resonance established by short pulses of laser energy (the laser is not part of the diagram). This transfers the particle's state "information" to the cavity (the exact process is QED-specific). A particle from B is then passed from R to R^{-1} through the cavity; the particle's state is changed to the exclusive-or of its initial state. R, R^{-1} and $R_z(\theta)$ are Ramsey zones. The Ramsey zones act as a sensor to let us know when the qubit has left the gate. Ramsey zones "sense" atomic oscillations and are named for Norman F.

Ramsey. This implementation meets the requirements of logical/physical reversibility, coherence and time independence. Entanglement would be achieved by correlating multiple cavities. Physical reversibility follows because if we take particle B and run it backwards through the gate we can change its state back to what it was. Logical reversibility follows because if we perform the XOR of A and B we will get what B was originally. Coherence is satisfied within the area of the gate if the gate is cooled sufficiently to exclude thermal variations and is properly shielded from outside sources of radiation (the cavity provides a degree of isolation). However, the loss of coherence between gates could be a problem. Time independence is met because the gate itself does not change with time and the system has been sufficiently shielded from external influences.

Several quantum-mechanical apparati have been proposed as possible implementations as quantum gates that do not completely meet the requirements for a quantum gate, as for instance:

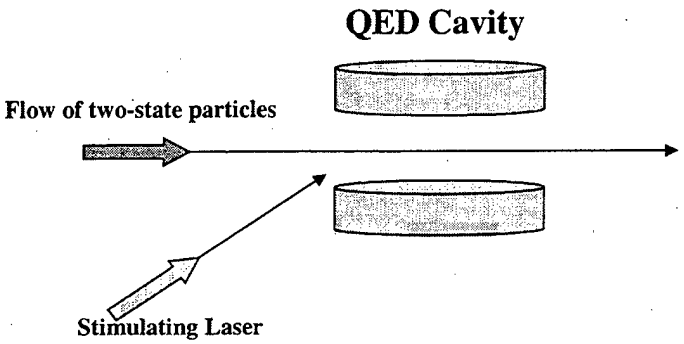


Figure 7

Here a flow of two-state particles, similar to the previous example, are passed through a QED cavity. When the stimulating laser is activated, the cavity resonance will flip the spin of the particle. The claim is if we know the spin of the particle before it goes into the cavity and the cavity is in resonance, then the output particle will have opposite spin to that of the input particle; this simulates an XOR gate. This model does comply with the stipulations of logical/physical reversibility, coherence and time independence. However, when we turn on the laser we are choosing which particles we flip the spin on. Then it cannot be a quantum process; if we know the spin that means we have somehow

measured it. Additionally, how would you chain these apparati together to do a meaningful quantum calculation? You cannot, because you won't know when to turn on the other stimulating lasers to activate the other cavities. If you did, it would not be a quantum process.

Not every developed "quantum device" could be a gate for a quantum computer. For example, a new quantum transistor developed at Sandia National Laboratories utilizes a quantum phenomenon of "tunneling" to achieve much faster transistors [Ref. 14]. This transistor called a DELTT (Double Electron Tunneling Transistor) has its switches closed all the time. The electrons pass to the other side of the switch by tunneling when their energy is great enough but still less than the potential energy required to surmount the open switch. According to classical mechanics a ball in a gravitational potential (say a bowl) cannot escape the confines of the bowl unless sufficient energy is imparted to it to have it roll over the lip of the bowl. However, in quantum mechanics there is a finite probability that the ball can "tunnel" through the side of the bowl and emerge on the other side with far less energy than is required to roll it out. Products in the commercial market exploit this phenomenon, like Charge-Coupled Device (CCD) cameras. Although this circuit element is a quantum device it could not be utilized as a quantum computer gate because it does not allow for reversibility: absorption destroys the quantum mechanical information about the particle.

C. QUANTUM WIRES

The last section has shown how difficult it is to build quantum devices which do not simply mimic a logic gate but which actually can be used as a building block for quantum computers. Once this problem is solved the gates need to be connected with quantum

wires. What constitutes a quantum wire varies with the implementation. However, they need to preserve the three conditions stipulated for quantum gates. One proposed implementation of quantum wires is a ballistic approach, which would shoot qubit particles between gates to achieve interconnection. This approach assumes a fairly high degree of shielding from the environment to prevent decoherence between quantum gates. Another approach is quantum teleportation. Quantum teleportation would "transport" a qubit from one gate and materialize it in or at the next gate. Quantum teleportation is explained later. As with quantum computation, a major hurdle in connecting quantum gates is decoherence; other problems include timing and stray particle interactions. Experimental confirmation of quantum wire implementations would provide a major step toward the realization of a quantum computer.

D. I/O PROBLEM

Another major hurdle that must be cleared before any quantum computer can be built is the I/O problem: How do you entangle more than two qubits at a time for input, how do you read the answer when the calculation is done, and how do you know when it is done?

Entanglement of more than two qubits at a time is very hard [Ref. 27]. We must find a source that emits three or more photons at a time when it transitions from a high energy state to a low energy state, and currently there are no known natural sources that do this. The second part of the I/O problem is: If you have classical data that has been successfully transformed into qubits and you pass these qubits through a quantum-gate array, then how will you know when the calculation is complete? You cannot measure it because you will collapse the wave function and fix it in a state that may or may not be

the answer you are looking for. A solution has been proposed which we will look at shortly. The third part of the I/O problem was considered in our general quantum computer model in Chapter III. The output interrogator must preserve the reversibility, coherence and time independence so we can extract the overall property in which we are interested, to harness the full quantum advantage.

Feynman proposed, as a solution to the second problem, that a cursor qubit be created to sample the data qubits. This cursor bit would travel from qubit-to-qubit to track the calculation. This cursor qubit could then be sampled without disturbing the original calculation, which allows a monitoring of a quantum process by proxy. However, there are several problems with this implementation. How can we entangle more than three qubits at a time? We can't. Let us suppose that we could entangle multiple particles simultaneously. If we measure the cursor qubit then we are measuring the system that is entangled; therefore we collapse the wave function and influence the calculation in a destructive manner. Let us now suppose that we isolate the cursor qubit from the system so when we measure it the wave function will not collapse. If we isolate the cursor qubit, it has no "knowledge" of the calculating system and cannot tell us when it is done. This clearly requires further work.

E. CHUANG'S PROOF OF CONCEPT

1998 saw the first experimental proof of concept for a quantum computer [Refs. 16, 17]. Prior to this it was theorized that the quantum coherence problem was too hard to solve and would prevent development of a quantum computer in the near term. The approach used Nuclear Magnetic Resonance (NMR) techniques on a solution of chloroform to perform a database search known as the Deutsch-Jozsa algorithm. The

Deutsch-Jozsa algorithm seeks to determine if a function is constant or balanced (having half the time a value of 0 and half the time a value of 1) in a shorter time than could be processed sequentially. To solve the problem for a function ϕ having a domain (input) covering N symbols, we must compute the function ϕ for N symbols to determine if ϕ is constant or balanced. On a classical sequential machine this requires N steps. A parallel classical computer could do those N steps simultaneously, provided it has N microprocessors. This is the worst case for both a deterministic (classical) machine and a classical stochastic machine [Ref. 24, p. 556]. A quantum computer on the other hand, will be able to harness quantum parallelism and accomplish this task in 1-2 steps by exploiting the orthogonal property of the superposed answer. When the quantum computation is finished and the answer is interrogated to be a 0, the function can be inferred to be constant because the answer is orthogonal to the original wave function (the inner product of two orthogonal vectors is 0). If the answer is interrogated to be a 1, then the function is balanced.

Although this experiment was not the most interesting application of quantum computing, it does provide a basis for future work. NMR/chloroform computing does not make for a practical computation machine because a large magnetic source (NMR) would have to be located close to a digital computer to provide the user interface, and chloroform is somewhat volatile with its state having to be checked frequently. Additionally, large computations would require 10 or more qubits and NMR computing does not provide the signal strength to distinguish between so many individual spins with a chloroform solution.

F. SUMMARY OF IMPLEMENTATION APPROACHES

The following table was compiled from references 1, 3 and 16. Each implementation provides its own model of quantum computation. The likelihood of success represents my judgement based on group discussions and literature readings (i.e., references, Internet etc.). Many implementations are variations on a theme (Optical Lattice, NMR etc.).

Table 2

Implementations	Interest (# of Articles)	Problems	Likelihood of Success (1-10)
Nuclear Magnetic Resonance (NMR)	20	1. Scalability 2. Limited number of qubits due to medium 3. Volatility of computing solution 4. Coherence times decrease for solutions of larger molecules	5
Ion Trap	24	1. Scalability 2. Decoherence due to thermal coupling 3. Speed limited to vibrational mode of ions 4. Limited number of qubits due to heating and coherence problems	4
Quantum Cavity (QED)	6	1. Spontaneous emission of atoms from the excited state (one form of decoherence) 2. Trapping and localization of atoms within cavities 3. Maintaining coherence between multiple cavities	7
Optical	2	1. Lack of interaction between photons (quantum interference) 2. Limited number of qubits (can only produce two entangled photons currently) 3. No nonlinear photon gates operating at the two-photon level	8
Optical Lattices		1. Scalability 2. Spontaneous emission within the	

	14	lattice (one form of decoherence) 3. Inability to affect only one photon/atom with laser energy 4. Scattering of laser energy 5. Thermal insulation	8
Silicon-Based Nuclear Spin (condensed matter form of NMR)	2	1. Nanoscale fabrication 2. Limitation of probability of error for each operation 3. Placing phosphorus atoms in a array in a semiconductor crystal 4. Development of defect-free semiconductor 5. Decoherence rate of phosphorus qubits	6
Quantum Dot	11	1. Scalability (quantum dot size not scalable smaller than current technology) 2. Decoherence times 3. Coherent optical control of quantum-dot states 4. Thermal insulation/isolation	6
Josephson Junction	9	1. Nanoscale fabrication 2. High-precision timing control 3. Residual two-qubit interactions 4. Decoherence times 5. Thermal insulation/isolation	4
SQUID (another form of Josephson Junction)	12	1. Inability to entangle multiple qubits 2. No demonstrated logic gates 3. Thermal insulation/isolation (operating temperature) 4. Junction quality (limit number of impurities) 5. Suppression of competing modes 6. Magnetic coupling of flux qubits to magnetic impurities 7. Small junction capacitances 8. Fabrication technology	3
Neutral Atom (extension of Optical Lattice)	1	1. Low number of neutral atoms in the lattice (limited number of qubits) 2. No method for addressing and reading out of qubits 3. Must implement quantum error correction methods 4. Investigate atomic collision	3

		effects (separations of atoms are small collisions most likely cause decoherence)	
Bose Condensate (in an optical lattice)	4	1. Thermal insulation/isolation (operating temperature) 2. Not enough experimental evidence to support model 3. Single particle theory, multiple particle experiment	2

Ion-trap implementation confines identical ions in a laser-cooled radio-frequency trap such that the ions reside in a one-dimensional lattice in a harmonic oscillation. Quantum-gate interactions are induced by stimulating the ions with a coherent pulse of laser energy to affect the center-of-mass mode of the collective quantum system. This one-dimensional lattice models a quantum register.

The cavity QED implementation utilizes the quantized mode of a high-finesse optical or microwave cavity to change the electronic states of atoms or photons. The cavity mode is stimulated into resonance or out of resonance by short pulses of laser energy.

Optical implementations of quantum computers are either linear or nonlinear. The linear optical implementation is a two-path state system that relies on a photon's propensity to take one of two possible paths upon exiting a beam-splitter. Gates are implemented by linear optical methods: beam splitters, mirrors, polarizers, wave plates etc. In a nonlinear optical implementation, gates are implemented by photonic qubits shifting the phase of others.

Implementations utilizing optical lattices are a condensed-matter variation of the optical method. Atoms are trapped in a lattice structure by lasers that perform cooling

and trapping. Here the atoms are bound together by light instead of a chemical bond like solids.

Silicon-based nuclear spin implementations are semiconductor-based NMR techniques that dope a silicon wafer with phosphorus. Metal electrode gates overlying the phosphorus-doped wafer perform logic-gate operations by affecting the shape of the dopant wave function.

Quantum-dot implementations use the two spin states of a single electron quantum dot. Logic gates are implemented by several means: reducing the gating voltage for the tunneling barrier to allow electrons to hop from dot to dot, influencing electron spin with another electron spin, influencing charge orbital degrees of freedom, and control of quantum-dot states by optical excitation.

A Josephson junction is a special switch that is composed of two superconductors sandwiching an insulating material. This switch has the property that current will flow without the application of a voltage potential. Current flow is produced by electron tunneling across a small insulating barrier [Ref. 26]. The two charge states of the superconducting island surrounding the junction would constitute a qubit. Logic gate operations are implemented by sequences of voltages across the junctions to vary the charge state. SQUID (Superconducting Quantum Interference Device) is a variation of the Josephson junction implementation where the presence of a quantized flux acts as the qubit to implement a two-state system.

Neutral-atom quantum computers are a variation of the optical lattice, where neutral atoms form the lattice and their electric dipoles function as qubits. Logic gates

are implemented by forcing two neutral atoms to the same potential to generate a dipole-dipole interaction.

A Bose-Einstein condensate (BEC) occurs when bosons (atoms with integer spin) are cooled to near absolute zero temperature and begin to act as one atom [Ref. 25]. Placing these condensate atoms in an optical lattice and moving optical-trap potentials so bosons move in between lattice sites would be an implementation of a BEC quantum computer.

V. A QUANTUM COMPUTER SIMULATOR

To analyze the possibilities of quantum computers we have three choices: Build analytical models, simulate those models, or measure actual systems. The last choice is not yet possible because there are no permanent working models. Analytical models can only tell us so much about the behavior. Therefore we are left with simulation to study quantum computation. Any simulation of a quantum computer should incorporate the elements discussed in Chapter III: superposition, entanglement, logical/physical reversibility, time independence and output interrogation. Reference 2 addressed the problem of simulating quantum effects on a discrete classical computer. It is hard to simulate quantum effects; we can, however, reproduce an approximation on a deterministic machine. Reference 18 provides a C++ version of a quantum-computer simulator that is programmed to run on a parallel machine (a machine with more than one microprocessor). This program provided the basis for a simulation we wrote. However, there is no one-to-one correspondence between reference 18 code and our simulator.

Java® was chosen as the programming language for the simulator. Java is more platform-independent than C++ and can be embedded in HTML Web pages, thus allowing more access to those curious about quantum computation. The Java program turned out to be less verbose than the C++ version in reference 18. Due to Java's strong typing, many functions that would have to be defined as class functions in C++ were simply written as member functions in Java (this saves the programmer from defining the functions) or were provided as part of the library accompanying the language. A quantum-computer simulator should simulate the workings of a quantum computer faithfully enough that we can make predictions of how an actual quantum computer

would behave and how we might be able to program it. A quantum computer should set up some input qubits into superposed and entangled states. These qubits should then pass through a series of quantum logic gates that must obey the precepts set forth in Chapter III. These logic gates should perform a kind of calculation on the qubits. After the calculation is done we should interrogate the values of the qubits and interpret our answer. This simulator implements portions of the model discussed in Chapter III, and is displayed schematically below.

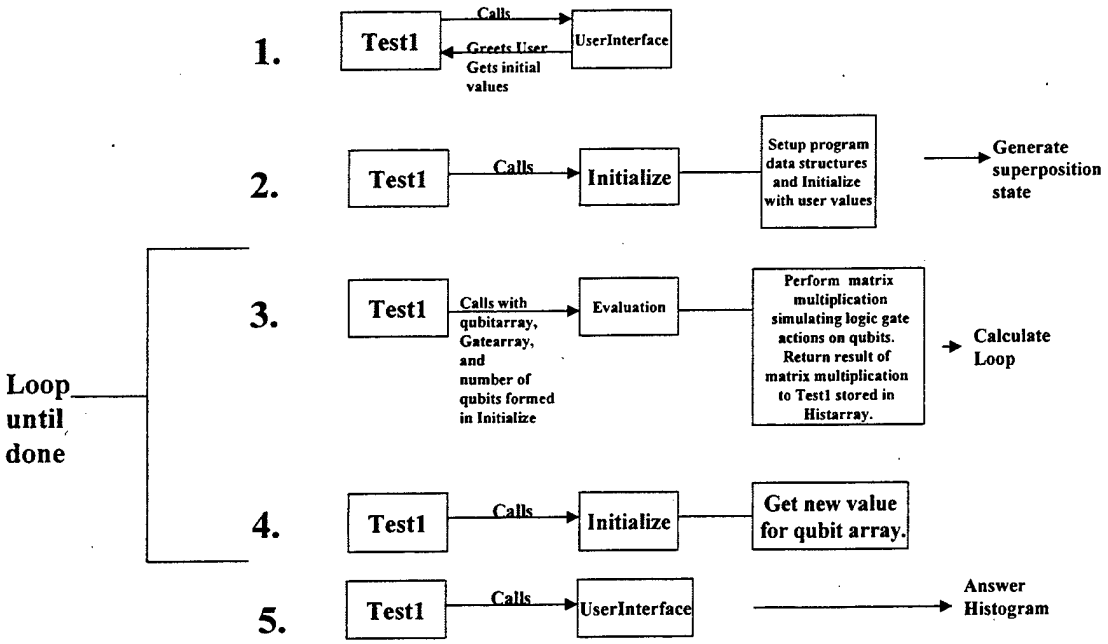


Figure 8

The simulator program is divided into 6 files/classes: Test1, UserInterface, Matrix, Complex, Initialize and Evaluation. Appendix A contains the source code.

Qubit superposition is realized in a literal way, by loading a data structure with 2^n possible values that the user input could be. These values are extracted from the data structure randomly, converted to binary, and the binary representation placed in a one dimensional array of size n. In a quantum system all values would take part in the calculation simultaneously; however because of the sequential nature of classical microprocessors, multiple random drawings from a data structure are implemented to simulate the quantum nature of the calculation. Qubit entanglement is not implemented. This process starts with the Test1 driver program calling UserInterface to collect the user input values. Test1 then calls Initialize to implement the superposition. Also, before calculation can begin, the simulator must initialize the quantum gate that will be used for calculation.

A Fredkin gate is used by the simulator to perform a quantum calculation. A gate data structure is set up as an N x N matrix where N is the number of elements in the one-dimensional array. The matrix representation for the gate is presented in reference 18 and is implemented with real and imaginary components which simulate the real and imaginary portions of the wave function $\Psi(t)$. Actual gate operation is simulated by the Evaluation class, which performs a matrix multiplication. Test1 performs a looped calculation which corresponds to incrementally stepping through all the possible values of the qubits. Currently the loop does 100 iterations maximum. Random possible values are withdrawn from the data structure, converted to binary, and calculated with the gate matrix.

When the calculation is complete, the answer is a one-dimensional mixed (real and complex) array that is parsed by Test1. Test1 acts as the output interrogator, directly

interpreting the results from the answer array. This is done by setting up a histogram array that is the same size as the answer array and parsing through the answer array until a nonzero element is encountered; whereupon the corresponding element in the histogram array is incremented by 1. For instance the output might look like "01010101" which is 85 in base 10.

The argument could be made that as the qubit values are withdrawn from the data structure and converted, they should be placed in an indeterminate state (by giving them a real and complex component) as they would in nature. However, this would add complexity to the matrix multiplication.

Because this simulation was kept as simple as possible, it does not provide essential services for the user and the fidelity required to study quantum computation. No useful calculation is performed by this simulation; it should be viewed as a proof of concept for implementation by a platform-independent language (Java). Entanglement is not implemented because it is a uniquely quantum effect. Entanglement might be implemented if an algorithm could be found to simulate the interaction amongst the qubits. Additionally, a GUI (Graphical User Interface), more selection of gates, and an ability to build a quantum circuits could be implemented. Hard-coding of the maximum number of iterations should be replaced by a fidelity selected by the user (high fidelity = 1,000 iterations, medium fidelity = 500, etc.).

VI. CAPABILITIES OF QUANTUM COMPUTERS

Quantum computers appear to be able to accomplish everything that a classical discrete computer can [Ref. 19]. Reference 19 shows that a quantum system can be used to emulate any Turing machine. Therefore, since Turing machines are universal and can do any conceivable computation, then a quantum system can too. But a quantum computer may not always perform as efficiently as a classical discrete computer. One reason is that Amdahl's Law says we can only increase processing speed by speeding up the slowest portion of a computation, and many sequential computations would work better on classical computers than on quantum computers. A second reason is that quantum systems are not ideal for all parallel problems since they are advantageous only for problems with low I/O costs and many solution constraints.

A. QUANTUM ALGORITHMS

Algorithms specifically designed for quantum computation are starting to appear. Two important ones are the Schor and Grover algorithms. The Schor algorithm provides a way to find the prime factorization of a large integer in polynomial time. The success of the Schor is based upon "modular periodicity, coupled with the power of the Fourier Transform for exposing periodicity" [Ref. 15, p. 18]. The Schor algorithm works by finding the periodicity of the input integer, which corresponds to the prime factor(s) of the input integer. Grover's algorithm provides a fast database search method which seeks to limit the number of accesses to a database. Grover's algorithm seeks to find an element x_0 from a database that contains many x by realizing the shortest path through the complex plane [Ref. 28]. The search uses parallel states of a quantum system to search

through all possible answers at once. Successful search paths reinforce one another while unsuccessful ones interfere randomly.

B. NP-HARD/COMPLETE PROBLEMS

Quantum computers may be good for solving NP-Hard/Complete problems. Roughly speaking, an NP-Hard/Complete problem is one where an answer is obtainable, but the time required to get the answer is prohibitively long. The class of these problems that are now efficiently solvable by current computing methods is rather small. P represents the class of problems that are solvable in effort that is a polynomial function of input size. Complexity class P problems are run on classical computers everyday and include sorting a list, searching a database, and most arithmetic operations. The class of NP-Hard/Complete problems do not have efficient algorithms to implement them, and take effort that is an exponential function of the input size. An example is the Knapsack problem which involves trying to pack items into a knapsack such that the sum of the items equals the capacity of the knapsack. Reference 15 suggests that quantum computers may be adept at solving such problems by checking all possible subsets in parallel. Something similar is done in the Schor algorithm which takes a large integer and finds all its prime factors by looking for periodicities.

However, several limitations need to be overcome before more general quantum algorithms can be considered. One is the lack of branching ability in a quantum system; one-way branching can check for a certain condition before proceeding with a program/algorithm. This is not possible in a quantum algorithm because it violates reversibility. Other limitations such as limited entangled states restrict the development of practical quantum algorithms.

C. QUATUM TELEPORTATION AND COMMUNICATION

By utilizing the quantum effect of nonlocality, qubit information can be transported from one quantum gate to another ("teleported") without losses due to decoherence or stray particle interactions. By utilizing quantum effects such as polarization, spin etc., it may be possible to encode information so you could always know if someone read it. This is quantum cryptography. Quantum communication uses similar quantum effects to provide communications across various media.

D. TECHNOLOGICAL FEASIBILITY

The following table summarizes the major area of quantum information systems.

Table 3

Quantum Technology	Interest (# of Articles)	Problems	Prob. Of Success (1-10) subjective
Computers	242	1. Scalability 2. Thermal isolation/insulation 3. Decoherence 4. Nanoscale fabrication 5. Single particle theory, multi-particle experiment 6. Connection of multiple gates (quantum wires) 7. Implementation of 'branching' (if …then) 8. Multi-particle entanglement	8
Teleportation	25	1. Decisionmaker and public acceptance of action at a distance 2. Decoherence 3. Limited amount of information that can be teleported	7

Cryptography	82	1. Decoherence 2. Multi-particle entanglement sources 3. Used for key distribution only (limited amount of information that can be transferred)	9
Communication	34	1. Decoherence 2. Multi-particle entanglement sources 3. Distance over which signals can be sent (30 Km) 4. Limited bandwidth	4

VII. POSSIBLE MILITARY APPLICATIONS

When considering military applications for quantum computing, we should consider computing problems in the military that current computers cannot solve or take too long to solve. Additional restrictions would include non-portability of the platform the quantum computer is mounted on and a consideration of Amdahl's Law; $T_c = T_s + T_p/N$ (T_c = total computation time, T_s = total serial computation time and T_p/N = total parallel computation time divided by the number of microprocessors).

Mobile applications of quantum computing are not possible in the near or mid-term. First, the required amount of shielding from the environment would prohibit deployment of the system to a harsh and unforgiving environment (i.e., desert, tropical jungle, etc.), which is where most military deployments take place. Second, the current and projected size of the systems will prohibit deployment in relatively small craft (i.e., tank, aircraft, etc.). Third, quantum computing is insufficiently reliable (real-time systems must have redundancy built-in because human life is at stake), and requires specialized repair staff (and education) and costly repair parts.

Given this restriction, military computing applications that might benefit from quantum computing are:

- -Intelligence
  - -Cryptography
    - -Increased Security
    - -Breaking Codes
    - -Secure Distribution of Data
  - -Photo Analysis
  - -Data Fusion
  - -Scan of EM Spectrum
- -Artificial Intelligence
- -Acquisitions
  - -Modeling and simulation
- -Game Theory

A. INTELLIGENCE

1. Cryptography

Reference 5 provides an experimental implementation of teleportation of secure information. An advantage of this method is that the information can be transmitted independent of the media used. This is due to a property of quantum mechanics known as non-locality. Two entangled photons A and B are created from the same source; we could destroy photon A and reconstitute its information from B by transferring it to another photon C. The amount of information that can be transferred is limited because it requires the entanglement of multiple particles, which is hard to do [Ref. 27].

Public-key encryption is popular and in wide use. The Schor algorithm on a quantum computer would allow a user to break the keys generated by a public-key scheme faster. This has obvious utility to the intelligence community.

2. Photo Analysis

Image processing algorithms typically look for repeating or periodic patterns in a photograph. This is very computationally intensive for a classical discrete system and current algorithms produce good results with certain pictures but not with others. Quantum systems may be more adept at solving this particular problem because they excel at finding the "periodicity" in functions. Images could be searched for known shapes (e.g., tank, aircraft), which are worthy of an analyst's scrutiny.

3. Data Fusion

All commanders desire a more comprehensive picture of the battlefield. Intelligence sections are tasked with assimilating a vast amount of data into a coherent

picture of the battlefield for the commander. This assimilation is a process whereby the intelligence officer reviews the combat information and sorts out what is and is not valid based on his or her experience. A quantum system could assist by sorting and sifting through the raw data looking for predetermined themes and concepts. A system utilizing Grover's algorithm could do this. This would reduce the man-hours spent looking through the raw data.

4. Scan of Electromagnetic Spectrum

Some intelligence systems scan for electromagnetic signals and determine bearing to the signal, signal strength, and what the signal might be from (e.g., commercial TV, radio). These systems are limited in how many and what types of signals can be intercepted. A quantum system with a high degree of parallelism might be able to process the data much faster, permitting analysis of multiple signals from multiple sources or spread-spectrum signals.

5. Secure Distribution of Data

Quantum communications (see Reference 1) could allow the secure exchange of information with one hundred percent assurance as guaranteed by Bell's inequality. If a source were to emit two entangled photons going in opposite directions and one photon is directed down an optical-fiber cable, then if an eavesdropper were to tap into the line, you would instantly know it. This is because the eavesdropper would have to absorb the original photon and could never reproduce the statistical nature of the original particle. Any reemitted photons would stand out against the naturally occurring error and loss rate associated with network communications. However, the impediments to this are listed in the last chapter. The most troubling impediment is the requirement that the photons be

entangled. Currently only two entangled photons can be produced from a single source and this would therefore limit practical application to secure key exchanges. Some researchers believe that other ways can be found to produce a higher bandwidth.

B. ARTIFICIAL INTELLIGENCE

Artificially intelligent quantum systems could be used by commanders as decision aids. They could assimilate the vast amounts of data that need to be considered and produce reasonable courses of action. A quantum system could allow a commander to vary parameters more broadly than they can today to see how different decisions would affect his or her command. With an artificially intelligent quantum system, personality traits of opponent commanders could be programmed into scenarios or general tactics of an opponent in a wargame. A quantum system could also be used as a training aid. Quantum parallelism could make it possible to speed up renditions of virtual environments and increase their possible realism.

C. ACQUISITIONS

Quantum systems could provide the same type of decision aid to program managers and program executive officers that it could provide to commanders. Quantum systems provide the potential to increase the fidelity of models and simulation through quantum parallelism and the knowledge inherent in them. With a quantum system we can encode more information about the problem due to superposition and entanglement. This increased information could permit more fidelity and allow modeling of more complex systems (like explosions) than current classical computers.

D. GAME THEORY

Game Theory is being used to study how groups of people interact [Ref. 7, p. 636]. Since this field is focussed on probability of outcome, it should mesh well with a quantum computation. In reference 6 the authors argue that since game theory is probability-based, it could exploit a superposition of possible outcomes that would be allowed by quantum mechanics. A popular game-theoretic model of the "Prisoner's Dilemma" is used to illustrate their point. Two prisoners are held in separate rooms so that they cannot communicate. The table below illustrates the advantages for each prisoner with the rows designating the benefits for prisoner #1 and the columns designating the benefits for prisoner #2 where the object is to maximize the score (based on a scale of 1-10, 10 being the maximum). For example, row 2 column 1 (10,0) represents prisoner #1 maximizing his score by confessing while prisoner #2 does not.

Table 4

	Prisoner 2 Does not Confess	Prisoner 2 Confess
Prisoner 1 Does not Confess	5,5	0,10
Prisoner 1 Confess	10,0	1,1

The best strategy when viewed by each prisoner alone is to "double cross" your partner. However, if they both do this it earns both of them a jail sentence and none of the money from the crime. So it is better to not double-cross. From a quantum mechanical perspective, we can avail ourselves of the superposition of states and remove the case of the Dilemma [Ref. 6]. This is done by assigning the classical strategies two base vectors $|A\rangle$ and $|B\rangle$, performing unitary transformations on them and realizing an

this solution is incomplete because there is no correlation to an actual realizable result. With more study meaningful methods could be developed to provide insight into and help in solving decision problems and dilemmas with a quantum approach.

VIII. CONCLUSION

There is great potential offered in quantum computation, although a paucity of algorithmic support at this time. By investigating it we gain insight into computation in general, and this may allow us other advances that are not yet realized. A quantum computer ought to be built soon. This will force direct investigation of troubling areas such as decoherence, scalability etc. Even a negative result would have significant scientific meaning. A few promising implementations could be funded with the purpose of developing a working prototype within the next ten years. However, the diverse funding advocated in reference 15 should be avoided. We also need to build a dictionary of common terms; computer scientists and physicists use terms with different meanings to each group.

Quantum cryptography was rated with the highest probability of success in Chapter VI because security is of high concern to business and governments. This is evidenced by the interest in the Schor algorithm. However, several hurdles need to be cleared before any serious implementation can occur.

Although quantum teleportation sounds far-fetched, it has been experimentally verified. However, wide implementations of this technology for communication are not likely because the maximum number of particles that we can currently entangle is two. This limits how much information that can be transmitted at one time. However, it could provide the answer to the "quantum wires" question: By teleporting qubits between gates, one could avoid decoherence and stray interactions. This topic requires further investigation.

Quantum communication currently faces stiff competition from current methods. The bandwidths proposed are very small and would only serve as a means of key exchange. Therefore problems such as multi-particle entanglement must be solved to make it practical.

LIST OF REFERENCES

1. Williams, C.P., Clearwater, S.H., *Explorations in Quantum Computing*, p. 8-9, Springer-Verlag, Inc., 1998.

2. Feynman, R.P., "Simulating Physics with Computers", *International Journal of Theoretical Physics*, vol. 21, no. 6/7, 1982.

3. Brandt, H.E., "Qubit Devices and the Issue of Quantum Decoherence", *Progress in Quantum Electronics*, vol. 22, no. 257-370, 1998.

4. Marand, C., Townsend, P.D., "Quantum Key Distribution over Distances as Long as 30 km", *Optics Letters*, vol. 20, no. 16, 1995.

5. Furusawa, A., "Unconditional Quantum Teleportation", *Science*, vol. 282, 1998.

6. Eisert, J., "Quantum Games and Quantum Strategies", http://xxx.lanl.gov/quant-ph/9806088, vol. 2, 1999.

7. Winston, W.L., *Operations Research: Applications and Algorithms*, p. 636-663, PWS-Kent, 1987.

8. Deutsch, D., "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer", *Proceedings of the Royal Society London*, A 400, 97-117, 1985.

9. Feynman, R.P., "Quantum Mechanical Computers", *Optics News*, February, 1985.

10. Hopcroft, J.E., Ullman, J.D., "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley, 1979.

11. Marshall, I., Zohar, D., "Who's Afraid of Schrodinger's Cat?", Bloomsbury, 1997.

12. Jammer, M., "The Philosophy of Quantum Mechanics", John Wiley&Sons, 1974.

13. Bennett, C., "Quantum Information and Computation", *Physics Today* , American Institute of Physics, p. 24-30, 1995.

14. Simmons, J., "Quantum Transistor May Lead to Much Faster Computers", http://www.sandia.gov/InsideSandia/IS05-98/quantum.html, 1998.

15. Kimble, H., "Quantum Computing", JASON JSR-95-115, Mitre Corp., 1996.

16. Chuang, I., "Experimental Realization of a Quantum Algorithm", http://squint.stanford.edu/qc/nmrdj/index.html, 1998.

17. Chuang, I., "Experimental Implementation of Fast Quantum Searching", http://squint.stanford.edu/qc/nmrqc-grover/index.html, 1998.

18. Marshall, J., "Simulating Quantum Circuits on a Parallel Machine", http://www.doc.ic.ac.uk/~jim1/report.html, 1996-97.

19. Benioff, P., "Quantum Mechanical Turing Machines That Dissipate No Energy", *Physical Review Letters*, vol. 48, no. 23, 1581-1585, 1982.

20. Cormen, T., "Introduction to Algorithms", MIT Press, 1990.

21. Joint Staff, "Joint Vision 2010", http://www.dtic.mil/doctrine/jv2010/jvpub.htm, 1994.

22. Barenco, A., "Elementary Gates for Quantum Computation", http://xxx.lanl.gov/PS_cache/quant-ph/pdf/9503/9503016.pdf, 1995.

23. Cleve, R., "Quantum Algorithms Revisited", http://xxx.lanl.gov/format/quant-ph/9708016, 1997.

24. Deutsch, D., "Rapid Solution of Problems by Quantum Computation", *Proceedings of the Royal Society London*, A 439, 553-558, 1992.

25. Moerdijk, A., "Bose-Einstein Condensation with Trapped Atoms", http://Herman.ni.phys.tue.nl/ardjan/bec.html, 1995.

26. Feynman, R., "The Feynman Lectures on Physics", Addison-Wesley, vol. III, 21-(14-18), 1966.

27. Baer, W., "Stanford University Site Visit", Inter-department memo, 1999.

28. Jozsa, R., "Searching in Grover's Algorithm", http://xxx.lanl.gov/quant-ph/990121, 1999.

APPENDIX A

```
//*****************************************************************
//
// Author:      Jack Mades
//
// Program:    Quantum Computer Simulator
//
// Date:        4 November 1998
// Compiler:   JDK 1.1.7
// Description:  This program will simulate a quantum computer.
// This will comprise the kernel portion of the quantum computer.
//
// Assumptions:  Probabilities will be 1 based (i.e. that random
// numbers > .5 represent a 1 and those < .5 will represent a 0).
// That pseudo random numbers generated by the computer represent
// the probability value associated with the wave function for the
// particle representing the qubit.
//
//*****************************************************************

public class Test1 {

   public static void main (String args[]){

    int numqubits = 0;   // number of qubits that user is working with
    int initvalue = 0;   // initial value that user wants to look for
    int temp = 0;
    int count = 0;
    int result = 0;

    Object newInt;

    String str = new String();

    Initialize init = new Initialize();

    UserInterface Uinter = new UserInterface();

    Uinter.Greeting();

    numqubits = Uinter.GetQubits();

    Evaluation eval = new Evaluation();
```

```
init.initarray(numqubits);

initvalue = Uinter.GetInitialValue();

System.out.println("\n\n Numqubits is " + numqubits);

init.initvector(numqubits);

init.initGatearray(numqubits);

int HistArray[] = new int[numqubits];

int tempArray[] = new int[numqubits];

init.toBinary(initvalue, numqubits);

eval.MatrixMultiply(init.qubitarray, init.Gatearray,
                numqubits);

while ( temp != 100) {

  count = (int)(Math.random() * ((numqubits-1) * 10));
  newInt = init.qubitvalues.elementAt(count);

  str = newInt.toString();

  try{
    result = Integer.parseInt(str);
    System.out.println("Next number in vector is " + result + "\n");
  }
  catch (Exception e){
    System.out.println("ParseInt failed in Test1 main \n");
  }
  init.toBinary(result, numqubits);
  eval.MatrixMultiply(init.qubitarray, init.Gatearray,
                numqubits);
  for (int iy = 0; iy < numqubits; iy++){
    if (eval.GetArray(iy) != 0 ) {
      HistArray[iy] += 1;
    }
  }
  temp++;
}
for (int iy = 0; iy < numqubits; iy++) {
   System.out.println(HistArray[iy]);
```

```
    }

  }
}
```

```
//*****************************************************************
//
// Author:      Jack Mades
//
// Program:    UserInterface
//
// Date:        4 November 1998
// Compiler:   JDK 1.1.7
//
//*****************************************************************

import java.io.*;

public class UserInterface {

  BufferedReader br;

  public void UserInterface(){
  }

  public void Greeting(){
    System.out.println("Welcome to the NPS Quantum Computer Simulator!\n\n");
  }

  public int GetQubits(){

    int numqubits = 0;
    String initInt;
    br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("How many qubits do you want to work with ? \n");
    try {
      initInt = br.readLine();
      numqubits = Integer.parseInt(initInt);
      System.out.println("Processing for " + numqubits + " begun...");
    } catch (Exception e) {
       System.out.println(e);
    }

    return numqubits;
  }

  public int GetInitialValue(){

    int initvalue = 0;
```

```
    br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Please enter the initial value for processing. \n");
    try {
      String initInt = br.readLine();
      initvalue = Integer.parseInt(initInt);
      System.out.println("Initial value of " + initvalue + " received.");
    } catch (Exception e) {
       System.out.println(e);
    }

    return initvalue;
  }
}
```

```
//*****************************************************************
//
// Author:      Jack Mades
//
// Program:    Evaluation
//
// Date:        4 November 1998
// Compiler:   JDK 1.1.7
// Description:
//
//*****************************************************************

import java.util.Vector;

public class Evaluation {

  Complex zerocomp = new Complex();
  Complex icomp1 = new Complex();
  Complex icomp2 = new Complex();

  Object outputVector[];

  int countArray[];

  public void Evaluation() {
  }

  public void MatrixMultiply(int initarray[], Object GateArray[][],
                      int matsize) {
    int rcomp = 1;
    int imgcmp = 1;
    int icmp = -1;

    Complex Gatecomp = new Complex();
    Complex Incomp = new Complex();

    //Create the output vector of the Matrix Multiplications
    outputVector = new Object[matsize];

    //Create the complex vector of the qubits
    Object compvec[] = new Object[matsize];

    icomp1.setComplex(rcomp, imgcmp);
    icomp2.setComplex(rcomp, icmp);
```

```
    for (int zy = 0; zy < matsize; zy++){
      if (initarray[zy] == 0){
        compvec[zy] = zerocomp;
      }else {
        compvec[zy] = icomp1;
      }
    }

    for (int ix = 0; ix < matsize; ix++) {
      for (int iy = 0; iy < matsize; iy++) {
        Gatecomp = (Complex)GateArray[ix][iy];
        if( Gatecomp == zerocomp ){
          outputVector[iy] = zerocomp;
        }
        else if( compvec[iy] == zerocomp ) {
          outputVector[iy] = zerocomp;
        }
        else {
          Incomp = (Complex)compvec[iy];
          outputVector[iy] = Incomp.multiply(Gatecomp);
        }
      }
    }

    CountOutput( outputVector, matsize);
  }

  public void CountOutput( Object Output[], int size ) {

    countArray = new int[size];
    Complex tmpcomp = new Complex();

    for (int zx = 0; zx < size; zx++ ) {
      tmpcomp = (Complex)(Output[zx]);
      if (tmpcomp == zerocomp){
        countArray[zx] = 0;
      }
      else {
        countArray[zx] = 1;
      }
    }
  }
  public int GetArray( int index ) {

  return countArray[index];
```

```
  }
}
//*********************************************************************
//
// Author:      Jack Mades
//
// Program:     Initialize
//
// Date:        4 November 1998
// Compiler:    JDK 1.1.7
// Description:
//
//*********************************************************************

import java.util.Vector;
import java.sql.Time;

public class Initialize {

  public Vector qubitvalues;
  public int qubitarray[];
  Complex zerocomp = new Complex();
  Complex icomp1 = new Complex();
  Complex icomp2 = new Complex();


  //Create the gate array to move the qubits through
  Object Gatearray[][];



  public void Initialize()  {
  }

  public void initarray(int numqubits){

    double value;

    qubitarray = new int[numqubits];

    System.out.println("Qubit Array Initialized !\n");

    Toffoli2Gate num1gate = new Toffoli2Gate();
  }

  public Double initQubitvalue(){
```

```
    Double vqbit = new Double(Math.random());

    return vqbit;
  }

  public void initvector (int numqubits){

    Time currtime = new Time(0);
    int size = (int)(Math.pow(2,numqubits));

    System.out.println("Size is   " + size);

    qubitvalues = new Vector(size);

    for ( int ix = 0; ix < size; ix++ ) {
      Integer vecint = new Integer(ix);
      qubitvalues.addElement(vecint);
    }
    System.out.println("Vector initialized !");
  }

  public void initGatearray(int matsize){

    Gatearray = new Object[matsize][matsize];

    int rcomp = 1;
    int imgcmp = 1;
    int icmp = -1;

    icomp1.setComplex(rcomp, imgcmp);
    icomp2.setComplex(rcomp, icmp);

    //Initialize the gate values
    for (int ix = 0; ix < matsize; ix++) {
      for (int iy = 0; iy < matsize; iy++) {
        if( ix == iy){
          Gatearray[ix][iy] = icomp1;
        }
        else if( iy == (ix+1)){
          Gatearray[ix][iy] = icomp2;
        }
        else if( ix == (iy+1)){
          Gatearray[ix][iy] = icomp2;
        }
        else {
          Gatearray[ix][iy] = zerocomp;
```

```
        }
      }
    }

  }
  public int toBinary (int convert, int size) {

    int result = 0;
    String str = new String();
    char tmpchar;
    Integer newInt = new Integer(0);

    str = newInt.toBinaryString(convert);

    System.out.println("This is the binary string  " + str);

    int max = str.length();
    int diff = size - max;
    System.out.println("Diff is  " + diff);
    System.out.println("Max is  " + max);

    if ( diff != 0 ) {
      for ( int zy = 0; zy < diff; zy++ ) {
        qubitarray[zy] = 0;
      }
      for (int ix = diff; ix < (max + diff); ix++) {
        tmpchar = (str.charAt(ix - diff));
        if ( tmpchar == '1' ) {
          qubitarray[ix] = 1;
        } else if ( tmpchar == '0' ) {
           qubitarray[ix] = 0;
        }
      }
    } else {
       for (int ix = 0; ix < max; ix++) {
         tmpchar = (str.charAt(ix));
         if ( tmpchar == '1' ) {
           qubitarray[ix] = 1;
         } else if ( tmpchar == '0' ) {
           qubitarray[ix] = 0;
         }
       }
    }
  return result;
  }
}
```

```
//*********************************************************************
//
// Author:      Unknown
//
// Program:     Complex
//
// Date:        4 November 1998
// Compiler:    JDK 1.1.7
// Description:
//
//*********************************************************************

public class Complex{

  public float real;
  public float img;

  public void Complex() {
    real = 0;
    img = 0;
  }

  public void setComplex ( int realComp, int realImg ){
     real = realComp;
     img = realImg;
  }

  public Complex add (Complex i) {
     Complex comp = new Complex();

     comp.real = real + i.real;
     comp.img = img + i.img;

     return comp;
  }

  public Complex subtract (Complex i) {
     Complex comp = new Complex();

     comp.real = real - i.real;
     comp.img = img - i.img;

     return comp;
  }

  public Complex multiply (Complex i) {
```

```
    Complex comp = new Complex();

      comp.real = real * i.real - img * i.img;
      comp.img =  real * i.img + img * i.real;

    return comp;
  }

  public int MagSquared (Complex i) {
    int result = -1;
  return result;
  }
}
```

```
//*********************************************************************
//
// Author:       Jack Mades
//
// Program:     Matrix
//
// Date:          4 November 1998
// Compiler:    JDK 1.1.7
// Description:
//
//*********************************************************************

public class Matrix{

  public void Matrix () {

  }

  public void makeMatrix (int dim) {
    Object array [][] = new Object[dim][dim];
  }

}
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Ste 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Director, Training and Education 1
   MCCDC, Code C46
   1019 Elliot Road
   Quantico, VA 22134-5027

4. Director, Marine Corps Research Center 2
   MCCDC, Code C40RC
   2040 Broadway Street
   Quantico, VA 22134-5107

5. Director, Studies and Analysis Division 1
   MCCDC, Code C45
   3300 Russell Road
   Quantico, VA 22134-5130

6. Marine Corps Representative 1
   Naval Postgraduate School
   Code 037, Bldg. 330, IN-116
   555 Dyer Road
   Monterey, CA 93940

7. Marine Corps Tactical Systems Support Activity 1
   Technical Advisory Branch
   Attn: Maj J. C. Cummiskey
   Box 555171
   Camp Pendleton, CA 92055-5080

8. Professor Neil C. Rowe, CS/Rp 2
   Naval Postgraduate School
   Monterey, CA 93943

9. Dr. Wolfgang Baer, CS/Ba 1
   Naval Postgraduate School
   Monterey, CA 93943

10. Captain John E. Mades 2
    639 South 22nd
    Terre Haute, IN 47803

11. Chairman, CS 1
Naval Postgraduate School
Monterey, CA 93943